Ultimate Wordpress and WooCommerce Deployment Guide (Really Advanced)

New VPS

Stats:

```
vCPU:   4
Memory: 8GB
Disk:   200GB SSD
OS: Ubuntu 20.04 LTS
```

Update and Upgrade

```
sudo apt update -y && sudo apt upgrade -y
```

first time login and setup

```
ssh root@<ip-address-of-new-vps>
```

create a new superuser

```
adduser newuser
usermod -aG sudo newuser
exit
```

Login with new super user

```
ssh newuser@<ip-address-of-new-vps>
```

setting timezone and automatic upgrades

```
sudo dpkg-reconfigure tzdata
sudo apt install unattended-upgrades
sudo dpkg-reconfigure unattended-upgrades

sudo nano /etc/apt/apt.conf.d/50unattended-upgrades
```

Add `Unattended-Upgrade::Automatic-Reboot-Time "03:30";` or turn it off so the server doesn't restart automatically after upgrades.

generating ssh keys

```
ssh-keygen
sudo chmod 700 ~/.ssh
```

print the newly generated key

```
cat ~/.ssh/id_rsa.pub
```

copy and paste it in authorised keys

```
nano ~/.ssh/authorized_keys
```

paste, save and exit.

disable root login

Find the line that reads `PermitRootLogin yes` and change it to `PermitRootLogin no`. Hit CTRL-O then CTRL-X to save the changes. In order for the changes to take affect you must restart the SSH service:

```
sudo service ssh restart
```

Configuring Network stack TCP buffers and states

buffer size = network capacity * round trip time

For example, if the ping time is 30 milliseconds and the network consists of 1G Ethernet then the buffers should be as follows:

.03 sec X (1024 Megabits) X (1/8)= 3.84 MegaBytes

```
sudo sysctl -w net.core.rmem_max=16777216
sudo sysctl -w net.ipv4.tcp_rmem='4096 87380 16777216'
sudo sysctl -w net.core.wmem_max=16777216
sudo sysctl -w net.ipv4.tcp_wmem='4096 16384 16777216'
```

use cubic model for tcp congestion control.

```
sudo sysctl -w net.ipv4.tcp_congestion_control=cubic
```

To make the TIME_WAIT state last for 40 seconds, decrease the value of net.ipv4.tcp_fin_timeout to 20 seconds:

```
sudo sysctl -w net.ipv4.tcp_fin_timeout = 20
```

Improve utilization of tcp sockets

```
sudo sysctl -w net.ipv4.tcp_tw_reuse = 1
```

Raising server Limits

The queue size

The TCP stack tries to process data packets as soon as they arrive. If the rate of processing is low, the arriving data packets get queued up. The kernel usually specifies a limit on the total number of packets that can be queued at the server. The value is specified by the `net.core.netdev_max_backlog` key:

```
sysctl net.core.netdev_max_backlog
```

net.core.netdev_max_backlog = 300

Increase the queue size to a large value, such as 10000:

```
sudo sysctl -w net.core.netdev_max_backlog=10000
```

The listen socket queue size

The OS kernel defines a limit on the listen socket queue size. The limit is specified by the value of the `net.core.somaxconn` key:

```
sysctl net.core.somaxconn
```

net.core.somaxconn = 128

Now, increase the queue size to a large value, such as 4096:

$ sudo sysctl -w net.core.somaxconn=4096

net.core.somaxconn = 4096

Half-opened connections

When the server accepts a connection, the connection waits for an acknowledgment from the client. Until that has happened, the connection is in a half-opened state. The OS kernel limits the total number of connections that can be in such a state. The server will drop new requests if the limits are exceeded. The limit is specified by the value of the `net.ipv4.tcp_max_syn_backlog` key:

```
sysctl net.ipv4.tcp_max_syn_backlog
```

net.ipv4.tcp_max_syn_backlog = 256

Increase the size to a large value, such as 2048:

```
sysctl -w net.ipv4.tcp_max_syn_backlog = 2048
```

net.ipv4.tcp_max_syn_backlog = 2048

Ephemeral ports

Ephemeral ports are the port numbers used by an operating system when an application opens a socket for communication. These ports are short-lived and are valid endpoints for the duration of the communication. The Linux kernel defines the ephemeral ports against the net.ipv4.ip_local_port_range key:

```
sysctl net.ipv4.ip_local_port_range
```

net.ipv4.ip_local_port_range = 32768 61000

The two values signify the minimum and maximum port values out of the total of 65,535 available ports on any system. These values may look adequately large, that is, 61000 - 32768 = 28232 is the number of available ports. It is important to note that 28,232 is the total number of available ports on the system. It does not turn out to be the number of concurrent connections that the server can serve.

As explained in the TCP states section, TCP will block sockets in the TIME_WAIT state with duration of MSL x 2. By default, the MSL is 60 seconds, which makes the TIME_WAIT period 120 seconds long. Thus, the server can only guarantee 28232/120 = 235 connections at any moment in time. If the server is acting as a proxy server, that is, it is serving content from upstream, then the number of connections will be half, that is, 235/2 = 117. Depending on your service and the load, this may not be a great number to look at!

> TIP The number of ports guaranteed by the server at any moment in time can be increased by modifying the MSL. If the MSL is 30 seconds, the TIME_WAIT state comes out at 60 seconds. The result is 28232/60 = 470 available ports at any moment in time.

The range can be modified by specifying the minimum and maximum ports against the `net.ipv4.ip_local_port_range` key:

```
sudo sysctl -w net.ipv4.ip_local_port_range='15000 65000'
```

net.ipv4.ip_local_port_range = 15000 65000

This makes a total of 50,000 ports for TCP socket use.

Open files

The kernel considers each opened socket as a file. It also imposes an upper bound on the total number of opened files. By default, the limit is set to 1,024 opened files:

```
ulimit -n
```

1024

Considering the total ephemeral socket range, this range is too low to serve the desired purpose. Under load, the limit may lead to socket failure with Too many opened files error messages in syslog.

The limits can be modified by changing the values in `/etc/security/limits.conf`. The file defines a soft limit and a hard limit against an item. Increase these values for the nofile item as an asterisk (*) with the user:

```
* soft nofile 64000
* hard nofile 64000
```

Setting up Uncomplicated firewall

```
sudo ufw allow ssh
sudo ufw allow http
sudo ufw allow https

sudo ufw enable
```

Installing and setting up fail2ban

```
sudo apt install fail2ban

sudo systemctl enable fail2ban
sudo systemctl start fail2ban
```

Installing a few libraries and essential packages

```
sudo apt install libxml2 libxslt-dev libgd-dev libbz2-dev libreadline-dev build-essential checkinstall autotools-dev git libpcre3-dev zlib1g-dev libssl-dev libxslt1-dev ca-certificates -y
```

adding trusted certs config to wget

```
sudo nano /etc/wgetrc
```

add to end:

```
ca_directory=/etc/ssl/certs
```

Installing PCRE

```
wget https://ftp.pcre.org/pub/pcre/pcre-8.44.tar.gz
tar -zxf pcre-8.44.tar.gz
cd pcre-8.44
./configure --prefix=/usr                       \
        --docdir=/usr/share/doc/pcre-8.44 \
        --enable-unicode-properties       \
        --enable-pcre16                   \
        --enable-pcre32                   \
        --enable-pcregrep-libz            \
        --enable-pcregrep-libbz2          \
        --enable-pcretest-libreadline     \
        --enable-jit                      \
        --disable-static                 &&
make
sudo make install
cd ..
```

Installing Zlib

```
wget http://zlib.net/zlib-1.2.11.tar.gz
tar -zxf zlib-1.2.11.tar.gz
cd zlib-1.2.11
./configure
```

```
make
sudo make install
cd ..
```

Installing openssl

```
wget https://www.openssl.org/source/openssl-1.1.1k.tar.gz
tar -zxf openssl-1.1.1k.tar.gz
cd openssl-1.1.1k
./config --prefix=/usr/local/ssl --openssldir=/usr/local/ssl shared zlib
make
make test
sudo make install
cd ..
cd /etc/ld.so.conf.d/
sudo nano openssl-1.1.1k.conf
```

```
/usr/local/ssl/lib
```

---

save and exit

```
sudo ldconfig -v
sudo mv /usr/bin/c_rehash /usr/bin/c_rehash.BEKUP
sudo mv /usr/bin/openssl /usr/bin/openssl.BEKUP
```

Add `:/usr/local/ssl/bin` to environment path

```
sudo nano /etc/environment
```

save and exit after adding.

Installing Nginx Mainline with extra modules

At the time of writing this post the mainline version of nginx is 1.19.10. You can use the latest version, but just check the changelog between the latest version and 1.19.0 for any major changes.

Advanced installation of NGINX with extra modules (The useful ones)

```
wget http://nginx.org/download/nginx-1.19.10.tar.gz
tar -xzvf nginx-1.19.10.tar.gz

cd nginx-1.19.10
```

Listing the configure options

```
./configure --help
```

Useful third-party modules:

keep a directory for third-party modules and download them into the directory.

```
cd ..
sudo mkdir modules-thirdparty
cd modules-thirdparty
```

The below modules are latest as of making this post, try getting the latest stable versions

1. ngx_brotli

   Adding brotli libraries

   ```
   sudo apt install brotli libbrotli-dev
   ```

   downloading and decompressing the ngx_brotli module

   ```
   sudo wget -O 'ngx_brotli-1.0.0.tar.gz'
  https://github.com/google/ngx_brotli/archive/refs/tags/v1.0.0rc.tar.gz
   sudo tar -xzvf ngx_brotli-1.0.0.tar.gz
   ```

2. ngx_cache_purge

   ```
   sudo wget -O 'ngx_cache_purge-2.3.0.tar.gz'
  https://github.com/FRiCKLE/ngx_cache_purge/archive/refs/tags/2.3.tar.gz
   sudo tar -xzvf ngx_cache_purge-2.3.0.tar.gz
   ```

3. ngx_http_geoip2_module

   Adding brotli libraries

```
sudo apt install libmaxminddb-dev libgeoip-dev -y
```

downloading and decompressing the ngx_http_geoip2_module

```
sudo wget -O 'ngx_http_geoip2_module-3.3.0.tar.gz'
https://github.com/leev/ngx_http_geoip2_module/archive/refs/tags/3.3.tar.gz
sudo tar -xzvf ngx_http_geoip2_module-3.3.0.tar.gz
```

Install geo ip update for refreshing the geoLite2 database

```
sudo add-apt-repository -y ppa:maxmind/ppa
sudo apt-get update
sudo apt-get install -y libmaxminddb-dev

sudo apt-get install -y geoipupdate
```

update geoipupdate conf with your id and generated key

```
sudo nano /etc/GeoIP.conf
```

```
# Please see https://dev.maxmind.com/geoip/geoipupdate/ for instructions
# on setting up geoipupdate, including information on how to download a
# pre-filled GeoIP.conf file.

# Enter your account ID and license key below. These are available from
# https://www.maxmind.com/en/my_license_key. If you are only using free
# GeoLite databases, you may leave the 0 values.
AccountID **0**
LicenseKey **000000000000**

# Enter the edition IDs of the databases you would like to update.
# Multiple edition IDs are separated by spaces.
#
# Include one or more of the following edition IDs:
# * GeoLite2-ASN - GeoLite 2 ASN
# * GeoLite2-City - GeoLite 2 City
# * GeoLite2-Country - GeoLite2 Country
EditionIDs GeoLite2-Country GeoLite2-City
```

Save and close

```
sudo geoipupdate -v
```

deploy a cron job for auto refresh of the data base

create a crontab

```
sudo crontab -e
```

add to end of file and add extra blank line

```
30 0 * * 6 /usr/bin/geoipupdate -v
```

---

for viewing crontab

```
sudo crontab -l
```

the databases are usually stored in

```
/var/lib/GeoIP/GeoLite2-Country.mmdb
/var/lib/GeoIP/GeoLite2-City.mmdb
```

Please note them down when you run `sudo geoipupdate -v`

4. incubator-pagespeed-ngx

First install a few libraries

```
sudo apt-get install build-essential zlib1g-dev libpcre3 libpcre3-dev unzip
uuid-dev
```

Get the module

```
sudo wget -O 'incubator-pagespeed-ngx-1.13.35.tar.gz'
https://github.com/apache/incubator-pagespeed-
```

```
ngx/archive/refs/tags/v1.13.35.2-stable.tar.gz
 sudo tar -xzvf incubator-pagespeed-ngx-1.13.35.tar.gz
```

A little extra configuration for this module

```
 cd incubator-pagespeed-ngx-1.13.35.2-stable/
 sudo wget https://dl.google.com/dl/page-speed/psol/1.13.35.2-x64.tar.gz
 sudo tar -xzvf 1.13.35.2-x64.tar.gz
```

we can go back to our home directory for compiling and installing the webp libraries and come back to the nginx source code directory to continue with adding the ngx_webp module.

```
 cd
```

5. ngx_webp

First install a few image related libraries

```
 sudo apt-get install libjpeg-dev libpng-dev libtiff-dev libgif-dev
```

downloading the source code of libwebp

```
 sudo wget -O 'libwebp-1.2.0.tar.gz'
https://storage.googleapis.com/downloads.webmproject.org/releases/webp/libwe
bp-1.2.0.tar.gz
 sudo tar -xzvf libwebp-1.2.0.tar.gz
```

Compiling the libwebp library

```
 cd libwebp-1.2.0/
 sudo ./configure
 sudo make
 sudo make install
```

With libwebp install we can go head and download our ngx_webp module in the modules folder

```
 cd ../nginx-1.19.10/modules-thirdparty/

 sudo wget -O 'ngx_webp.zip'
https://github.com/vladbondarenko/ngx_webp/archive/refs/heads/master.zip
```

```
sudo unzip ngx_webp.zip
```

Small tweaking required

```
sudo nano ngx_webp-master/src/ngx_http_webp_module.c
```

Line 63

change from `pid_t parent_pid;` to `pid_t parent_pid __attribute__((unused));`

with that setup we can move back to the nginx source directory

```
cd ..
```

Finally we are ready to configure and compile our custom nginx source code. Feel free to enable or disable the below modules or add other modules to fit your need/use-case.

```
sudo ./configure --prefix=/etc/nginx \
--sbin-path=/usr/sbin/nginx \
--modules-path=/etc/nginx/modules \
--conf-path=/etc/nginx/nginx.conf \
--error-log-path=/var/log/nginx/error.log \
--http-log-path=/var/log/nginx/access.log \
--pid-path=/var/run/nginx.pid \
--lock-path=/var/run/nginx.lock \
--http-client-body-temp-path=/var/cache/nginx/client_temp \
--http-proxy-temp-path=/var/cache/nginx/proxy_temp \
--http-fastcgi-temp-path=/var/cache/nginx/fastcgi_temp \
--http-uwsgi-temp-path=/var/cache/nginx/uwsgi_temp \
--http-scgi-temp-path=/var/cache/nginx/scgi_temp \
--user=www-data \
--group=www-data \
--with-file-aio \
--with-threads \
--with-http_xslt_module=dynamic \
--with-http_image_filter_module=dynamic \
--with-http_geoip_module=dynamic \
--with-http_addition_module \
--with-http_auth_request_module \
--with-http_dav_module \
--with-http_flv_module \
--with-http_gunzip_module \
--with-http_gzip_static_module \
--with-http_mp4_module \
--with-http_random_index_module \
```

```
--with-http_realip_module \
--with-http_secure_link_module \
--with-http_slice_module \
--with-http_ssl_module \
--with-http_stub_status_module \
--with-http_sub_module \
--with-http_v2_module \
--with-mail \
--with-mail_ssl_module \
--without-mail_pop3_module \
--with-stream \
--with-stream_realip_module \
--with-stream_ssl_module \
--with-stream_geoip_module=dynamic \
--with-stream_ssl_preread_module \
--with-pcre=../pcre-8.44 \
--with-pcre-jit \
--with-zlib=../zlib-1.2.11 \
--with-compat \
--add-module=./modules-thirdparty/ngx_webp-master \
--add-dynamic-module=./modules-thirdparty/ngx_brotli-1.0.0rc \
--add-dynamic-module=./modules-thirdparty/ngx_cache_purge-2.3 \
--add-dynamic-module=./modules-thirdparty/ngx_http_geoip2_module-3.3 \
--add-dynamic-module=./modules-thirdparty/incubator-pagespeed-ngx-1.13.35.2-stable
\
--with-cc=/usr/bin/gcc \
--with-cc-opt='-g -O2 -fPIE -fstack-protector-strong -Wformat -Werror=format-
security -fPIC -Wdate-time -D_FORTIFY_SOURCE=2' \
--with-ld-opt='-Wl,-Bsymbolic-functions -fPIE -pie -Wl,-z,relro -Wl,-z,now -fPIC -
static-libstdc++'
```

Compilation with turning off warnings to errors of unused variables

```
sudo make
```

Installation with turning off warnings to errors of unused variables

```
sudo make install
```

A few more steps

```
cd /usr/sbin
sudo ln -s /usr/share/nginx/sbin/nginx nginx
sudo mkdir /usr/share/nginx/
cd /usr/share/nginx/
```

```
sudo ln -s /etc/nginx/modules modules
sudo mkdir -p /var/lib/nginx/body
```

Adding nginx as a systemd service

```
sudo nano /lib/systemd/system/nginx.service
```

```
[Unit]
Description=The NGINX HTTP and reverse proxy server
After=syslog.target network.target remote-fs.target nss-lookup.target

[Service]
Type=forking
PIDFile=/run/nginx.pid
ExecStartPre=/usr/sbin/nginx -t
ExecStart=/usr/sbin/nginx
ExecReload=/usr/sbin/nginx -s reload
ExecStop=/bin/kill -s QUIT $MAINPID
PrivateTmp=true

[Install]
WantedBy=multi-user.target
```

---

reload systemd

```
sudo systemctl daemon-reload
```

Installing and setting up PHP and PHP-FPM

```
sudo apt install software-properties-common

sudo add-apt-repository ppa:ondrej/php

sudo add-apt-repository ppa:ondrej/nginx-mainline

sudo apt update

sudo apt install php8.0

sudo apt install php8.0-fpm php8.0-common php8.0-cli php8.0-dev php8.0-imap
php8.0-soap php8.0-redis php8.0-xmlrpc php8.0-pdo php8.0-mysql php8.0-zip php8.0-
```

```
mbstring php8.0-curl php8.0-xml php8.0-bcmath php8.0-imagick php8.0-gd

sudo nano /etc/php/8.0/fpm/pool.d/www.conf
```

make sure the following is set

```
user = www-data
group = www-data
...
listen.owner = www-data
listen.group = www-data
...
pm.max_children = 20
pm.start_servers = 8
pm.max_spare_servers = 15
pm.process_idle_timeout = 10s;
pm.max_requests = 500;
```

```
sudo nano /etc/php/8.0/fpm/php.ini
```

```
upload_max_filesize = 64M
...
post_max_size = 64M
...
realpath_cache_size = 64M
...
opcache.force_restart_timeout=0
opcache.memory_consumption=512
opcache.interned_strings_buffer = 32
opcache.max_accelerated_files = 16000
opcache.optimization_level = 0xFFFFFFFF
```

```
sudo php-fpm8.0 -t

sudo service php8.0-fpm restart

php-fpm8.0 -v

sudo systemctl enable php8.0-fpm
```

Installing and setting up MySQL

```
sudo apt install mysql-server

sudo mysql_secure_installation
```

Set passwrd policy to 2, create new password and say yes to everything else.

```
mysql -u root -p

CREATE USER 'user'@'%' IDENTIFIED BY '************';
```

For remote administrator - not needed ---

```
GRANT ALL PRIVILEGES ON *.* TO 'user'@'%' WITH GRANT OPTION;

FLUSH PRIVILEGES;
```

```
CREATE USER 'wordpress'@'localhost' IDENTIFIED BY '************';

CREATE SCHEMA `wordpress` ;

GRANT ALL PRIVILEGES ON wordpress.* TO 'wordpress'@'localhost';
```

Tuning MySQL

```
sudo nano /etc/mysql/mysql.conf.d/mysqld.conf
```

Make sure the following are set

```
bind-address            = 0.0.0.0
key_buffer_size         = 32M
max_connections         = 512
# innodb_buffer_pool_chunk_size default value is 128M
innodb_buffer_pool_chunk_size=128M

innodb_buffer_pool_instances = 32
```

```
# innodb_buffer_pool_chunk_size * innodb_buffer_pool_instances
innodb_buffer_pool_size = 4096M

innodb_flush_method = O_DIRECT
innodb_log_file_size = 512M
innodb_flush_log_at_trx_commit = 2
thread_cache_size = 16

#query_cache_type = ON
#query_cache_size = 128M
#query_cache_limit = 256K
#query_cache_min_res_unit = 2k

max_heap_table_size = 64M
tmp_table_size = 64M
```

```
sudo mysql -t
sudo systemclt restart mysql
sudo systemclt enable mysql
```

Installing and setting up Redis

```
sudo apt install redis-server
```

Configure redis server for unix socket connection.

```
sudo usermod -a -G www-data redis

sudo nano /etc/redis/redis.conf
```

uncomment the following lines and change the unix socket permission to 770

```
unixsocket /var/run/redis/redis-server.sock
unixsocketperm 770
```

---

restart redis

```
sudo systemctl restart redis-server
```

Configuring Nginx

Adding SSL with cloudflare

```
sudo nano /etc/ssl/certs/cloudflare_example.com.pem
```

Paste the generated certificate preferably ECC certificate on cloudflare SSL > Origin Certificates > Create Certificate

```
sudo nano /etc/ssl/private/cloudflare_example.com.pem
```

Paste the generated private key here.

```
sudo nano /etc/ssl/certs/cloudflare_origin_ecc.pem
```

download Origin pull certificate from cloudflare

```
cd /etc/ssl/certs/
sudo wget https://support.cloudflare.com/hc/en-
us/article_attachments/360044928032/origin-pull-ca.pem

sudo openssl dhparam -out /etc/ssl/certs/dhparam.pem 4096
```

Nginx configuration

```
cd /etc/nginx/
sudo nano nginx.conf
```

```
# run nginx as www-data user
user www-data;

# Load modules
load_module ./modules/ngx_http_brotli_filter_module.so;
load_module ./modules/ngx_http_brotli_static_module.so;
load_module ./modules/ngx_http_geoip2_module.so;
```

```
load_module ./modules/ngx_stream_geoip2_module.so;
load_module ./modules/ngx_http_image_filter_module.so;
load_module ./modules/ngx_pagespeed.so;


# let nginx use 4 cpus available
worker_processes 4;

pid /run/nginx.pid;

#load_module;

worker_rlimit_nofile 16384;

# Events section
events {
    # worker connections = worker_rlimit_nofile/2
    worker_connections 8192;
    # use epoll for linux
    use epoll;
    # accept multiple requests
    multi_accept on;
    # accept_mutex keep it off unless required
    accept_mutex off;
    #accept_mutex_delay 500ms;
}

http {
    # tell nginx the different kinds of file types.
    include mime.types;
    default_type  application/octet-stream;
    # turn server tokens off for security
    server_tokens off;

    # Switch off all logs
    error_log /var/log/nginx/error.log crit;
    access_log off;
    # log_format combined '$remote_addr - $remote_user [$time_local] '
    #                     '"$request" $status $body_bytes_sent '
    #                     '"$http_referer" "$http_user_agent"';
    log_not_found off;

    # Switch off ability to show port number on redirect
    port_in_redirect off;

    # Allow more requests for keep-alive connection
    # keepalive_requests 100;

    # effecient sending of files - use sendfile for files below 4MB and then start
using     Async direct I/O for files bigger than 4 MB
    sendfile on;
```

```
    directio 4m;
    directio_alignment 512;
    aio on;

    # TCP settings
    tcp_nodelay on;
    tcp_nopush off;

    # Buffers
    client_body_buffer_size 16k;
    client_header_buffer_size 4k;
    large_client_header_buffers 8 16k;
    client_max_body_size 32m;
    client_body_in_single_buffer on;
    client_body_temp_path temp_files 1 2;

    # Keep alive
    keepalive_requests 100;
    keepalive_disable msie6;
    keepalive_timeout 30s 28s;

    # Timeouts
    lingering_close on;
    lingering_timeout 5s;

    resolver_timeout 10s;

    client_header_timeout 12s;
    client_body_timeout 12s;
    send_timeout 6s;

    # Enable reset connections on timeout
    reset_timedout_connection on;

    # File Caching - static content
    open_file_cache max=10000 inactive=20s;
    open_file_cache_valid 30s;
    open_file_cache_min_uses 2;
    open_file_cache_errors on;

    # Brotli configuration
    brotli on;
    brotli_comp_level 4;
    brotli_static on;
    brotli_types  application/atom+xml application/geo+json application/javascript
                  application/x-javascript application/json application/ld+json
                  application/manifest+json application/rdf+xml
application/rss+xml
                  application/xhtml+xml application/xml font/eot font/otf font/ttf
                  image/svg+xml text/css text/javascript text/plain text/xml;
```

```
    # Pagespeed Configuration
    pagespeed on;
    pagespeed RewriteLevel OptimizeForBandwidth;
    pagespeed FileCachePath "/var/cache/ngx_pagespeed/";
    pagespeed EnableFilters
combine_css,combine_javascript,convert_to_webp_lossless,    extend_cache;

#make_google_analytics_async,make_show_ads_async,collapse_whitespace,extend_cache,
extend_cache_pdfs,lazyload_images;

    # Include server configurations
    include conf.d/*.conf;
}
```

```
sudo nano conf.d/example.com.conf
```

```
server {
    listen 443 http2;
    server_name national-honey.com;

    # SSL certificates
    ssl_certificate /etc/ssl/certs/cloudflare_national-honey.com.pem;
    ssl_certificate_key /etc/ssl/private/cloudflare_national-honey.com.pem;

    # SSL settings
    include snippets/ssl-settings.conf;

    location / {
        root html;
        index index.html index.htm;
    }

    location ~ ".pagespeed.([a-z].)?[a-z]{2}.[^.]{10}.[^.]+" {
        add_header "" "";
    }

    location ~ "^/pagespeed_static/" { }
    location ~ "^/ngx_pagespeed_beacon$" { }

    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   html;
    }
}
```

```
server {
    listen 443 http2;
    server_name www.national-honey.com;
    return 301 https://$host$request_uri;
}

server {
    listen 80;
    server_name national-honey.com www.national-honey.com;
    return 301 https://$host$request_uri;
}
```

```
sudo nano snippets/ssl-settings.conf
```

```
# Origin Pull CA verification by Cloudflare
ssl_client_certificate /etc/ssl/certs/origin-pull-ca.pem;
ssl_verify_client on;

# Cloudflare CA Root cerificate verification
ssl_trusted_certificate /etc/ssl/certs/cloudflare_origin_ecc.pem;

# SSL settings
ssl_protocols TLSv1.3 TLSv1.2;
ssl_prefer_server_ciphers on;
ssl_ciphers "EECDH+AESGCM:EDH+AESGCM:AES256+EECDH:AES256+EDH";
ssl_ecdh_curve secp384r1;
ssl_session_cache shared:SSL:10m;
ssl_session_timeout 1h;
ssl_session_tickets off;
ssl_stapling on;
ssl_stapling_verify on;
resolver 8.8.8.8 8.8.4.4 valid=300s;
resolver_timeout 5s;

# SSL related Headers
add_header Strict-Transport-Security "max-age=63072000; includeSubdomains;
preload";
add_header X-Frame-Options sameorigin;
add_header X-Content-Type-Options nosniff;
add_header X-Xss-Protection "1; mode=block";

ssl_dhparam /etc/ssl/certs/dhparam.pem;
```

Adding SSL without cloudflare

//TODO: Add SSL configuration of Lets Encrypt with automatic renewal of certificates

Installing wordpress and setting up

```
sudo mkdir /var/www/wordpress
sudo chmod 777 -R /var/www/wordpress
sudo chown www-data:www-data -R /var/www/wordpress
cd /var/www/

sudo wget https://wordpress.org/latest.tar.gz

sudo tar -xzvf latest.tar.gz

sudo chown www-data:www-data -R /var/www/wordpress
sudo chmod 755 -R /var/www/wordpress

sudo nano wp-config-sample.php
```

Edit to suite your configuration

```
<?php
define('WP_CACHE', true);
// ** this has to be added to the start of the wp-config.php ** //

...

// ** MySQL settings - You can get this info from your web host ** //
/** The name of the database for WordPress */
define( 'DB_NAME', 'wordpress' );

/** MySQL database username */
define( 'DB_USER', 'wordpress' );

/** MySQL database password */
define( 'DB_PASSWORD', '********password_here********' );

/** MySQL hostname - using scoket connection */
define( 'DB_HOST', ':/var/run/mysqld/mysqld.sock' );

/** Database Charset to use in creating database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The Database Collate type. Don't change this if in doubt. */
define( 'DB_COLLATE', '' );

// ** this can be added after the MySQL config **//
/* Redis */
```

```
/** REDIS SOCKET */
define( 'WP_REDIS_SCHEME', 'unix' );

/** REDIS PATH TO SOCKET */
define( 'WP_REDIS_PATH', '/var/run/redis/redis-server.sock');

/**#@+
 * Authentication Unique Keys and Salts.
 *
 * Change these to different unique phrases!
 * You can generate these using the {@link https://api.wordpress.org/secret-
key/1.1/salt/ WordPress.org secret-key service}
* You can change these at any point in time to invalidate all existing cookies.
This will force all users to have to log in again.
 *
 * @since 2.6.0
 */
 // ** first line crutial for redis object caching ** //
define('WP_CACHE_KEY_SALT', 'example.com');
define('AUTH_KEY',         '**************');
define('SECURE_AUTH_KEY',  '**************');
define('LOGGED_IN_KEY',    '**************');
define('NONCE_KEY',        '**************');
define('AUTH_SALT',        '**************');
define('SECURE_AUTH_SALT', '**************');
define('LOGGED_IN_SALT',   '**************');
define('NONCE_SALT',       '**************');
```

---

generate salts and hashes at https://api.wordpress.org/secret-key/1.1/salt/

and paste it in it's place.

```
sudo mv wp-config-sample.php wp-config.php
```